Detecting Windows Operating System's Ransomware based on Statistical Analysis of Application Programming Interfaces (API) Functions Calls

Abdulgader Almutairi

Assistant Professor, College of Sciences and Arts at Qassim University – Kingdom of Saudi Arabia (KSA)
azmtierie@qu.edu.sa

***Abstract*-Malicious software, or malware in short, pose a serious threat to and can severely damage computer systems. A prime example of this was a ransomware that widely exploited a number of systems recently. Ransomware is a dangerous malware, which is used for extorting money and ransom from victims, failing which they could lose their data forever. In this paper, we used statistical analysis of Application Programming Interfaces (API) functions calls in order to detect ransomware adequately. First, we imported API functions calls for numerous ransomware and benign software applications samples, and saved them as strings in strings files. Subsequently, the imported API functions calls were counted and tabulated to generate a dataset. Then, we applied Chi-Square, Paired Sample t-test, and Correlation statistical analysis to our generated dataset. Our statistical analysis was able to detect ransomware effectively with almost 95% accuracy. In addition, we determined the relationship between each pair.**

I. Introduction

Nowadays, computer systems simplify and organize the performing of most day to day tasks in the modern life. Unfortunately, these computer systems are threatened by malicious and hostile software applications called malwares. [1][2] "Malware" is for a portmanteau of "malicious software application," which negatively impacts computer systems. [1][3] It became one of the major cyber threats currently and can perform malicious actions, including espionage, information stealing.

In this era of computers, malwares consist of viruses, worms, and trojans, and recently, ransomware. [4] A virus is a malicious software that injects itself into a carrier program in order to deliver harmful and undesired functions. [2][5][4] A worm is similar to a virus, except that it propagates itself through computer networks. [6] A trojan is a malicious software that looks legitimate but it is not, and it causes damage and opens backdoors to attackers. A ransomware is a malicious software that prevents or limits the owners from accessing their data until they pay a ransom. [7][8][9]

Recently, ransomwares were the cause of huge monetary losses including $5B worldwide in 2017 alone, which is expected to reach $11.5B by 2019.Therefore, extreme caution should be taken in order to protect computer systems and networks. [7]

II. Literature Review

In the area of malware detection, numerous research works have been conducted based on Application Programming Interfaces (API) functions calls, but they have not adequately covered ransomware.

A. *Related Research Works*

A research work in [10] analyzed imported API functions calls in order to detect the malware, but it did not cover encrypt and decrypt API functions calls, which are crucial in ransomware. Another research in [11] applied C 4.5 decision tree algorithm with n-grams=6 to API system calls to detect anomalies, and therefore to secure virtual machines. The study in [11] did not cover ransomware. An additional research work in [12] applied various machine learning algorithms like DT, SVM, and Random Forest to differentiate malware from cleanware. The study in [12] collected various malware samples, but not ransomware. Similarly, study in [13] applied various machine learning techniques to detect malware, but did not show ransomware in the collected analyzed samples. A research work in [4] applied DT machine learning algorithm to several inputs that were collected from various sources like System, Disk, Network, Registry, and Browser. Like the past studies, this study did not apply DT machine learning algorithm to several inputs caused by ransomware. A research in [14] analyzed malware based on binary file features after converting it into an image file in order to analyze it. The weakness of this study is that an attacker can play with bits of binary file, which yields a change in the image file and therefore it will fail to detect malware. One more research work in [15] applied SVM machine learning algorithm to its own generated dataset of Windows API with various values of n-grams, but it did not show ransomware samples among the collected samples. A study in [16] applied machine learning algorithms like DT and SVM to text mining of API. It used t-test statistical distribution to test the significant difference between them. The study in [16] did not cover encrypt and decrypt API functions calls, which are used widely by ransomware. Another research work in [17] used Markov Chain to analyze API in order to detect malware. Like the previous researches, this study did not cover encrypt and decrypt API functions calls, which are used widely by ransomware. A research in [18] used Z statistical distribution to analyze Linux KVM system calls in order to detect anomalies. As well, this research did not cover encrypt and decrypt API functions calls, which are used widely by ransomware.

III. Generating Dataset for Windows Operating Systems Ransomware and Benign Software Applications based on their API Functions Calls

Firstly, we collected numerous ransomware and benign software applications samples for Windows Operating System in order to perform and execute the statistical analysis. The names of collected ransomware software applications samples are Locky.exe, Petya.exe, Cerber.exe, TeslaCrypt.exe, Vipasana.exe, WannaCry.exe, Mamba.exe, Petrwrap, and WannaCry_Plus. The names of collected benign software applications samples names are calc.exe, cliconfg.exe, clipbrd.exe, explorer.exe, freecell.exe, notepas.exe, taskmgr.exe, and spider.exe. The ransomware software applications samples were collected from theZoo [19] and reverse [20] websites, whereas the benign software applications samples were collected from Win systems32 directory.

Secondly, all software applications samples—whether ransomware or benign—were exported into strings throughout the following executable command under Linux operating system terminal: **$ strings softwareName.exe >> softwareName.exe.str** where the softwareName.exe involves all ransomware and benign software application samples that are listed above. After that, we studied and analyzed the exported strings software applications samples due to Windows Operating System Application Programming Interfaces (API) functions calls according to the following Win API functions calls families:

1. Internet Connection
2. Shell Code
3. Encrypt, Decrypt and Digital Certificate
4. Embedded Resource
5. Devices
6. Process Thread Mutex Service and Event (Task)
7. Privileges and User
8. Files and Directories
9. Strings Manipulation
10. Registry and Memory Manipulation

The studying and analyzing of exported strings software applications samples due to Windows Operating System Application Programming Interfaces (API) functions calls identify and count API functions calls for each software application sample by using the following executable command under Linux operating system terminal: $ **cat softwareName.exe.str | grep -i "Keyword" | wc –l**, where Keyword is one of the ten API functions calls families that are listed above. Then, the results are presented in Table 1 below.

TABLE 1
WINDOWS OPERATING SYSTEM API FUNCTIONS CALLS FOR RANSOMWARE AND BENIGN SOFTWARE APPLICATIONS SAMPLES.

Software / Win API Functions Calls	Internet Connection	Shell Code	Encrypt, Decrypt, and Digital Certificate	Embedded Resource	Devices	Process, Thread, Mutex, Service, and Event (Task)	Privileges and User	Files and Directories	Strings Manipulation	Registry and Memory Manipulation	Class
1. Locky.exe	0	0	1	0	0	14	5	5	13	19	Ransomware
2. Petya.exe	17	1	30	15	5	88	20	135	122	70	Ransomware
3. Cerber.exe	5	2	14	6	1	42	9	42	40	22	Ransomware
4. TeslaCrypt.exe	6	3	11	1	1	25	3	28	44	7	Ransomware
5. Vipasana.exe	4	2	3	9	1	26	2	47	46	15	Ransomware
6. WannaCry.exe	0	0	8	4	0	11	4	27	10	10	Ransomware
7. Mamba.exe	0	5	65	48	68	292	57	234	186	79	Ransomware
8. calc.exe	0	0	0	0	0	8	3	3	10	4	Benign
9. cliconfg.exe	0	0	0	0	0	4	3	3	0	3	Benign
10. clipbrd.exe	0	0	0	0	2	22	10	33	23	15	Benign
11. explorer.exe	0	1	0	0	3	37	7	20	37	45	Benign
12. freecell.exe	0	0	0	0	1	2	3	2	4	9	Benign
13. notepas.exe	0	0	0	0	1	8	5	17	10	10	Benign
14. taskmgr.exe	0	0	0	0	2	23	6	2	17	13	Benign
15. Petrwrap	0	0	21	13	1	27	4	58	17	6	Ransomware
16. WannaCry_Plus	3	0	11	20	0	36	31	47	20	14	Ransomware
17. spider.exe	0	0	0	0	2	11	4	7	15	11	Benign

IV. Statistical Analysis of Dataset for Windows Operating Systems Ransomware and Benign Software Applications based on their API Functions Calls

In this research paper, we used Chi-Square statistical distribution to study and analyze the relationships between the ransomware and benign software applications, and Windows operating system API functions calls. The null hypothesis ($\mathbf{H_0}$) and alternative hypothesis ($\mathbf{H_1}$) to be studied and analyzed according to Chi-Square distribution in (1) are as follows:

H_0: There is no relationship between ransomware and benign software applications, and Win API functions calls.

H_1: There is a relationship between ransomware and benign software applications, and Win API functions calls.

$$\aleph^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \text{ .... } (1)$$

The calculated results of Chi-Square distribution for inputs in Table 1 are recorded in Table 2 below. As results show, only **Encrypt/Decrypt/DigitalCert** Win API functions calls family rejects the null hypothesis ($\mathbf{H_0}$) and accepts alternative hypothesis ($\mathbf{H_1}$), which indicates there is a relationship between ransomware and benign software applications, and Win API functions calls, since P-value is 0.03, which is less than 0.05 ($0.03 \leq 0.05$). This clearly stated that there is a 95% chance of detecting ransomware software applications once the software application uses Encrypt/Decrypt/DigitalCert Win API functions calls. On the contrary, all other Win API functions calls families failed to detect a ransomware software application once the software application uses its corresponding Win API functions calls family in chance 95%.

TABLE 2
CHI-SQUARE VALUES (P) FOR WIN API FUNCTIONS CALLS FAMILIES AND CLASS.

No.	Win API Functions Calls Family	Value	df	Asymp. Sig.
1	InternetConnection	6.3	5	0.278
2	ShellCode	4.78	4	0.311
3	Encrypt/Decrypt/DigitalCert	17	8	0.03
4	EmbeddedResource	13.43	8	0.098
5	Devices	6.83	5	0.233
6	Process/Thread/Mutex/Service/Event	14.99	14	0.379
7	Privileged/User	9.31	10	0.503
8	Files/Directories	17	13	0.199
9	Strings	12.32	13	0.502
10	Registry/Memory	12.99	14	0.528

In addition, Paired Sample t-test statistical distribution is used to study and analyze whether the mean difference between two sets of Win API functions calls families is zero or not. The null hypothesis ($\mathbf{H_0}$) and alternative hypothesis (**H₁**) to be studied and analyzed according to Paired Sample t-test statistical distribution based on equation in (2) and significance level α = 0.05 as follow:

$$\mathbf{H_0}: \mu_0 = \mu_1$$

$$\mathbf{H_1}: \mu_0 \neq \mu_1$$

$$t = \frac{m}{s/\sqrt{n}}, df = n - 1 \quad \text{...} \quad \mathbf{(2)}$$

Besides that, Correlation (r) is calculated for sets of Win API functions calls families according to equation (3) as follows:

$$r_{xy} = \frac{n\sum xy - \sum x \sum y}{\sqrt{n\sum x^2 - (\sum x)^2}\sqrt{n\sum y^2 - (\sum y)^2}} \quad \text{...} \quad \mathbf{(3)}$$

The calculated results of Paired Sample t-test statistical distribution and Correlation for Table 1 inputs are recorded in Table 3 below. As results show, P-value for the pair **EncryptDecryptDigitalCert – ProcessThreadMutexServiceEvent** is 0.03, which is less than 0.05, therefore we accepted the null hypothesis (**H_0**) and rejected alternative hypothesis (**H_1**). This indicates that the mean difference between the two sets of Win API functions calls families is zero (means are equal). In the same way, the pair has very strong positive correlation, since r = 0.94. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. Likewise, P-value for the pair **PrivilegedUser – FilesDirectories** is 0.01, which is less than 0.05, therefore we accepted the null hypothesis (**H_0**) and rejected alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is zero (means are equal). In addition, the pair has very strong positive correlation, since r = 0.87. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. Similarly, P-value for the pair **Devices – RegistryMemory** is zero, which is less than 0.05, therefore we accepted the null hypothesis (**H_0**) and rejected alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is zero (means are equal). In addition, the pair has strong positive correlation, since r = 0.72. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. Similarly, P-value for the pair **PrivilegedUser – Devices** is 0.02, which is less than 0.05, therefore we accepted the null hypothesis (**H_0**) and rejected alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is zero (means are equal). In addition, the pair has very strong positive correlation, since r = 0.86. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. Moreover, P-value for the pair **EmbeddedResource – ShellCode** is 0.05, which is less than or equal 0.05, therefore we accepted the null hypothesis (**H_0**) and rejected alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is zero (means are equal). In addition, the pair has medium positive correlation, since r = 0.68. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. Likewise, P-value for the pair **RegistryMemory – Strings** is 0.04, which is less than 0.05, therefore we accepted the null hypothesis (**H_0**) and rejected alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is zero (means are equal). In addition, the pair has very strong positive correlation, since r = 0.92. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. In contrast, P-value for the pair

InternetConnection – ShellCode is 0.25, which is greater than 0.05, therefore we rejected the null hypothesis (**H_0**) and accepted alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is not zero (means are not equal). In addition, the pair has low positive correlation, since r = **0.24**. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair. Similarly, P-value for the pair **FilesDirectories – ProcessThreadMutexServiceEvent** is 0.72, which is greater than 0.05, therefore we rejected the null hypothesis (**H_0**) and accepted alternative hypothesis (**H_1**). This indicates that the mean difference between two sets of Win API functions calls families is not zero (means are not equal). In addition, the pair has very strong positive correlation, since r = 0.94. This depicts that the increase in one family of one member of the pair will increase the other member of the same pair.

TABLE 3

A PAIRED SAMPLE T-TEST STATISTICAL DISTRIBUTION FOR THE MEAN DIFFERENCE BETWEEN TWO SETS OF WIN API FUNCTIONS CALLS FAMILIES.

Pair	t	df	Sig.	Correlation (r)
EncryptDecryptDigitalCert - ProcessThreadMutexServiceEvent	-2.36	16	0.03	0.94
PrivilegedUser - FilesDirectories	-2.73	16	0.01	0.87
InternetConnection - ShellCode	1.2	16	0.25	0.24
Devices - RegistryMemory	-4.1	16	0	0.72
PrivilegedUser - Devices	2.56	16	0.02	0.86
FilesDirectories - ProcessThreadMutexServiceEvent	0.36	16	0.72	0.94
EmbeddedResource - ShellCode	2.17	16	0.05	0.68
RegistryMemory - Strings	-2.21	16	0.04	0.92

V. CONCLUSION

Ransomware is the most hazardous malicious software (malware); therefore, they should be detected to secure computer systems. In this paper, we used statistical analysis of Application Programming Interfaces (API) functions calls in order to detect ransomware sufficiently. We applied Chi-Square, Paired Sample t-test, and Correlation statistical analysis to our generated dataset, which we produced from various ransomware and benign software applications. Our statistical analysis is able to detect ransomware effectively with an accuracy of 95%, especially when the software application uses Encrypt/Decrypt/DigitalCert Win API functions calls family. In addition, we stated that the mean difference between two sets of Win API functions calls families is zero for all pairs, except InternetConnection – ShellCode and FilesDirectories – ProcessThreadMutexServiceEvent. Similarly, correlation for all pairs is positive, which varies between Low, Medium, or Strong. In future work, we plan to extend this study to apply machine learning algorithms for further improvements.

REFERENCES

[1] A. Jadhav, D. Vidyarthi, and M. Hemavathy, “Evolution of evasive malwares: A survey,” *2016 Int. Conf. Comput. Tech. Inf. Commun. Technol. ICCTICT 2016 - Proc.*, pp. 641–646, 2016.

[2] H. T. Poon and A. Miri, “Scanning for Viruses on Encrypted Cloud Storage,” *Proc. - 13th IEEE Int. Conf. Ubiquitous Intell. Comput. 13th IEEE Int. Conf. Adv. Trust. Comput. 16th IEEE Int. Conf. Scalable Comput. Commun. IEEE Int.*, pp. 954–959, 2017.

[3] S. Anil and R. Remya, “A hybrid method based on genetic algorithm, self-organised feature map, and support vector machine for better

network anomaly detection," *Comput. Commun. Netw. Technol. (ICCCNT),2013 Fourth Int. Conf.*, pp. 1–5, 2013.

[4] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, "Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts," *Proc. - IEEE Symp. Secur. Priv.*, pp. 1009–1024, 2017.

[5] A. A. Shaikh, "Attacks on cloud computing and its countermeasures," *Int. Conf. Signal Process. Commun. Power Embed. Syst. SCOPES 2016 - Proc.*, pp. 748–752, 2017.

[6] F. C. C. Osorio, H. Qiu, and A. Arrott, "Segmented sandboxing - A novel approach to Malware polymorphism detection," *2015 10th Int. Conf. Malicious Unwanted Software, MALWARE 2015*, pp. 59–68, 2016.

[7] S. Morgan, "Ransomware Damage Report," 2017. [Online]. Available: https://cybersecurityventures.com/ransomware-damage-report-2017-part-2/. [Accessed: 16-Mar-2018].

[8] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," *Proc. 2004 ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '04*, vol. 7, p. 470, 2004.

[9] P. Mishra, E. S. Pilli, V. Varadharajant, and U. Tupakula, "NvCloudIDS: A security architecture to detect intrusions at network and virtualization layer in cloud environment," *2016 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2016*, pp. 56–62, 2016.

[10] M. Belaoued and S. Mazouzi, "Statistical study of imported APIs by PE type malware," *Proc. - 2014 Int. Conf. Adv. Netw. Distrib. Syst. Appl. INDS 2014*, pp. 82–86, 2014.

[11] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Securing virtual machines from anomalies using program-behavior analysis in cloud environment," *Proc. - 18th IEEE Int. Conf. High Perform. Comput. Commun. 14th IEEE Int. Conf. Smart City 2nd IEEE Int. Conf. Data Sci. Syst. HPCC/SmartCity/DSS 2016*, no. Vmi, pp. 991–998, 2017.

[12] R. Tian, R. Islam, L. Batten, and S. Versteeg, "Differentiating malware from cleanware using behavioural analysis," *Proc. 5th IEEE Int. Conf. Malicious Unwanted Software, Malware 2010*, pp. 23–30, 2010.

[13] I. Firdausi, C. lim, A. Erwin, and A. S. Nugroho, "Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection," *2010 Second Int. Conf. Adv. Comput. Control. Telecommun. Technol.*, pp. 201–203, 2010.

[14] X. Han, J. Sun, W. Qu, and X. Yao, "Distributed malware detection based on binary file features in cloud computing environment," *26th Chinese Control Decis. Conf. (2014 CCDC)*, no. 4, pp. 4083–4088, 2014.

[15] R. Veeramani and N. Rai, "Windows API based Malware Detection and Framework Analysis," *... Conf. Networks Cyber Secur.*, vol. 3, no. 3, pp. 1–6, 2012.

[16] G. G. Sundarkumar, V. Ravi, I. Nwogu, and V. Govindaraju, "Malware detection via API calls, topic models and machine learning," *IEEE Int. Conf. Autom. Sci. Eng.*, vol. 2015–Octob, pp. 1212–1217, 2015.

[17] H. L. Hussein, "Static Analysis Based Behavioral API for Malware Detection using Markov Chain," vol. 5, no. 12, pp. 55–64, 2014.

[18] S. S. Alariti and S. D. Wolthusen, "Detecting Anomalies In IaaS Environments Through Virtual Machine Host System Call Analysis," *2012 Int. Conf. Internet Technol. Secur. Trans.*, pp. 211–218, 2012.

[19] Y. Nativ, "theZoo." [Online]. Available: https://github.com/ytisf/theZoo/tree/master/malwares. [Accessed: 15-Mar-2018].

[20] Reverse, "reverse." [Online]. Available: https://www.reverse.it/. [Accessed: 15-Mar-2018].